Shen *et al. EURASIP Journal on Wireless Communications and Networking* (2015) 2015:124
DOI 10.1186/s13638-015-0347-9

EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

C2EM: cloud-assisted complex event monitoring in wireless multimedia sensor networks

Hang Shen[1,2], Guangwei Bai[1,2,3*], Ding Ma[1], Lu Zhao[2] and Zhenmin Tang[2]

Abstract

This paper addresses the problem of how to provide a more flexible service for complex event monitoring in wireless multimedia sensor networks (WMSNs). In particular, we propose C2EM, a cloud-assisted complex event monitoring architecture that involves scalar sensors, camera sensors, cloudlets, and clouds to leverage computation offloading reliability, service response time, coverage efficiency, and energy efficiency. On clouds, we design an opportunistic service access point selection scheme that provides quality of service (QoS) supports for scalar sensor computation offloading. Meanwhile, clouds are responsible for optimizing camera sensor utilization of the whole network. On cloudlets, we design a real-time camera actuation scheme with the objective of minimizing the possible coverage overlaps while providing probabilistic guarantee in residual energy. Through computation division, most complex computations and network environment profilers are executed on cloudlets or clouds. Sensors only need to carry out very simple operations. We evaluate the performance of C2EM through simulations under a complex event scenario. The results demonstrate that C2EM can enhance complex event monitoring performance with optimized energy efficiency, desirable event coverage quality, and potential adaptability to the dynamics of complex events.

Keywords: Wireless multimedia sensor networks; Complex event monitoring; Cloud; Cloudlet; Computation offloading

1 Introduction

Wireless multimedia sensor networks (WMSNs) are an emerging type of sensor networks that contain a certain number of camera sensors in conjunction with a large number of scalar sensors and can collect and process multimedia data [1] under a limited supply of resources in computation, energy, bandwidth, and storage. Generally, scalar sensors are responsible for event detection, and camera sensors are used for event capturing in the form of pictures or video. A typical application of such networks is multimedia event monitoring. It is worth noting that the status (including region size and position) of many real-world events often changes over time, such as migration, traffic and crowd movement, and sudden hot spots. We refer to these events as complex events. The dynamic event characteristics lead to considerable computation cost, control message overhead, and data redundancy, resulting in significant performance degradation in both energy efficiency and monitoring quality. Many existing solutions are not scalable and not robust enough to adapt to complex event monitoring.

To achieve a more flexible service for complex event monitoring in WMSNs, apart from the primary goal of energy efficiency, quality of service (QoS) provisioning is also an important issue in system design. Many recent works have been proposed for enhancing event monitoring performance, including multimedia coding [2-4], event coverage [5-7], and event detection [8-10]. For multimedia coding-related works, as it relates to the transmission and exchange of large amounts of data, in addition to computation-intensive features, it also has resource-

*Correspondence: bai@njtech.edu.cn
[1]College of Electronics and Information Engineering, Nanjing Tech University, No. 30, PuZhu Road (South), Pukou District, 211816 Nanjing, China
[2]College of Computer Science and Engineering, Nanjing University of Science and Technology, Xiaolingwei Street 200, Xuanwu District, 210094 Nanjing, China
Full list of author information is available at the end of the article

demanding features. Even if computing power bottleneck could be alleviated, it can only obtain a limited performance improvement. In contrast, event coverage and detection require a smaller data exchange, while at the same time there are a large number of computation tasks. As a result, there is great potential to significantly improve overall performance of such applications if computing power bottleneck can be solved.

As an emerging computing mode, mobile cloud computing (MCC) [11], due to scalable cloud environment, offers an opportunity to extend resource-constrained wireless device capabilities for computation-intensive and energy-hungry applications by offloading of jobs that take place from itself to the cloud. In the last few years, various cloud-assisted solutions have been proposed, such as cloudlet [12], cloud torrent [13], CloneCloud [14], virtual cloud [15], and code offload [16]. Recent studies demonstrate that, compared with traditional service models, it provides benefits such as mobile computation speedup [17], energy efficiency [18], resource allocation [19], and QoS/QoE provisioning [20]. However, most of existing works focus on improving the experience of mobile smartphone users. In addition, although a number of solutions [21-25] regarding how to integrate wireless sensor networks with the clouds have been proposed for better performance, they cannot support complex event monitoring.

Intuitively, the computation cost on sensors can be reduced greatly if sensors can leverage MCC resources. In fact, the combination of WMSN and MCC for complex event monitoring is challenged by a series of special technical problems. At first, it is difficult to integrate complex software modules that support computation offloading into resource-constrained sensors, which is different from a smartphone or laptop. Moreover, even if it can be achieved, the dynamical cloud connections contribute negative effects to computation offloading reliability and service response time. These characteristics identify significant challenges for enhancing complex event coverage efficiency in WMSNs by leveraging MCC.

In this paper, we propose a cloud-assisted complex event monitoring (C2EM) architecture, which involves a scalar sensor tier, camera sensor tier, cloudlet tier, and cloud tier. Each tier interacts with one another, considering computation offloading reliability, service response time, coverage efficiency, and energy efficiency. C2EM has several advantages compared with previous approaches. First of all, it reduces sensor computation cost and message exchange complexity in the monitoring process. In addition, it enhances camera sensor utilization of the whole network, while considering real-time network performance. In other words, C2EM integrates the advantages of both distributed and centralized coverage approaches. The main contributions of this paper are summarized as follows.

1. A novel cloud-assisted architecture called C2EM is proposed, different from existing approaches that only rely on sensors. Most complex computation tasks are offloaded on clouds or cloudlets, and these sensors are only needed to carry out very simple operations, thereby reducing energy consumption.
2. The offloading reliability, service response time, event coverage efficiency, and energy efficiency are taken into account. Through two-level optimization based on cloudlets and clouds, C2EM can provide a more flexible event monitoring service that ignores the dynamic characteristics of a complex event.
3. We design an opportunistic service access selection scheme to provide QoS supports for scalar sensor computation offloading, building upon which an energy-aware camera actuation scheme is designed on cloudlets to minimize the possible coverage overlaps while providing probabilistic residual energy guarantee. Meanwhile, clouds are responsible for optimizing camera sensor utilization, by using event detection data stored in database servers, focusing on minimizing redundant multimedia data.
4. We evaluate the performance of C2EM through simulations under a complex event scenario. The simulation results demonstrate that C2EM can provide efficient complex event monitoring service with optimized energy efficiency, desirable coverage quality, and potential adaptability to the changes in network environment.

The remainder of this paper is organized as follows. Section 2 introduces the background of cloud-assisted methods and our motivation. In Section 3, we present a high-level overview of C2EM architecture. We then detail the important components of C2EM in Section 4. Section 5 presents simulation results that demonstrate the benefits of C2EM. Finally, we conclude this paper and outline some perspectives for our future work in Section 6.

2 Design challenges

We believe that cloud-assisted architecture can decisively contribute to event monitoring performance improvement. To design such an efficient monitoring system, we must address the following challenges:

1. Complex event monitoring is computation-intensive. These computations, however, if executed locally on sensors, usually demand a large quantity of CPU cycles, thus making network lifetime short. By deploying cloudlet access points (CAPs) manually or

utilizing ubiquitous service access points (if existing) to servers through wireless networks, computation on sensors can now be offloaded partially or completely to local cloudlets, thus saving energy while achieving desirable response time. To support computation offloading and result data feedback, some specific software modules, such as controller, profiler, and solver [26,27], are needed. However, it is not feasible to integrate all of these modules into sensors. As a result, it is necessary to consider how to reduce the complexity of computation offloading.

2. Inherent from wireless communications, MCC is also characterized by limited and time-varying bandwidth and by unstable and dynamic changes of wireless channel condition [28]. For these midway offloading computations, if a cloud connection is interrupted, more energy will be consumed due to re-execution of offloading. On the other hand, for these offloaded computations, once disconnected from the network, reconnection will prolong service response time. Using a general-purpose MCC to support complex event monitoring in WMSNs may suffer from unacceptable network performance. Therefore, we need to consider how to provide QoS supports in delay and reliability.
3. Local cloudlets have the advantage in terms of service response time, but they have less cloud storage and computing capacity server. On the other hand, although remote clouds have the advantage of storage space and computing power, the main disadvantage is the longer service response. Therefore, how to conduct a reasonable computation task division between clouds and cloudlets is the key to overall performance improvement.

3 Systems architecture

3.1 Network model

We assume that a set of scalar sensors, camera sensors, and CAPs are placed together randomly or manually throughout an area of interest to monitor various complex events. Denote by Φ the monitored region. All sensors and CAPs have fixed random position and can dynamically adjust their transmission radius. The position can be obtained from one of the techniques from [29]. Each scalar sensor has a fixed sensing radius. Denote by s_i the scalar sensor of ID i and ψ_i the sensing region of s_i. Each camera sensor has a certain field of view (FoV) and a fixed orientation, and it does not move. Denote by C the deployed CAP set. Denote by c_j the camera sensor of ID j and a_k the cloudlet access point of ID k. Denote by f_j the FoV and e_j the residual energy of c_j. Assume that each directed link has an error probability. The probability of direct delivery from a node of ID i to a node of ID j is given by p_{ij}.

Define x_j as the random variable to indicate the status of camera sensor c_j.

$$x_j = \begin{cases} 1, c_j \text{ is actuated,} \\ 0, c_j \text{ is turned off.} \end{cases} \quad (1)$$

When a camera sensor is turned off, it can only receive control messages. Once a camera sensor is actuated (turned on), it generates video streams transmitted to the convergence point.

3.2 Service model

As shown in Figure 1, C2EM consists of four tiers, i.e., scalar sensors, local cloudlets, camera sensors, and remote clouds. The relationship between the various tiers is summarized below.

1. Scalar sensors are used to detect possible events. When an event takes place, a scalar sensor will detect the event and offload computation along with its own ID to the local cloudlets through CAPs until they cannot detect these events.
2. When a computation task arrives, the result computed by cloudlets will be sent to camera sensors through CAPs in the form of an actuation control message.
3. Each cloudlet continuously synchronizes stored data with clouds. Clouds are able to query cloudlets for real-time network environment factors through the Internet.
4. Clouds are responsible for scalar sensor offloading to find suitable CAPs. In addition, clouds continuously adjust camera status throughout the monitoring region, using the event detection data stored in database servers, focusing on minimizing the possible coverage overlaps, and resulting in the elimination of redundant data.
5. When receiving a control message from cloudlets or clouds, camera sensors adjust their status in accordance with specific rules.

3.3 System model

Figure 2 provides a high-level overview of the C2EM architecture. The most complex computation tasks (including CAP selection, event coverage control, and event data analysis) and continuous environment profilers (including available bandwidth, energy consumption, and server workload) are executed on cloudlets and clouds. We transfer the responsibility of camera control to cloudlets through CAPs, which usually have lower latency and are more energy efficient. C2EM uses event detection data stored in database servers on clouds to optimize event coverage in terms of camera sensor utilization as well as energy efficiency. As we can see, the camera sensors, cloudlets, and clouds will exchange

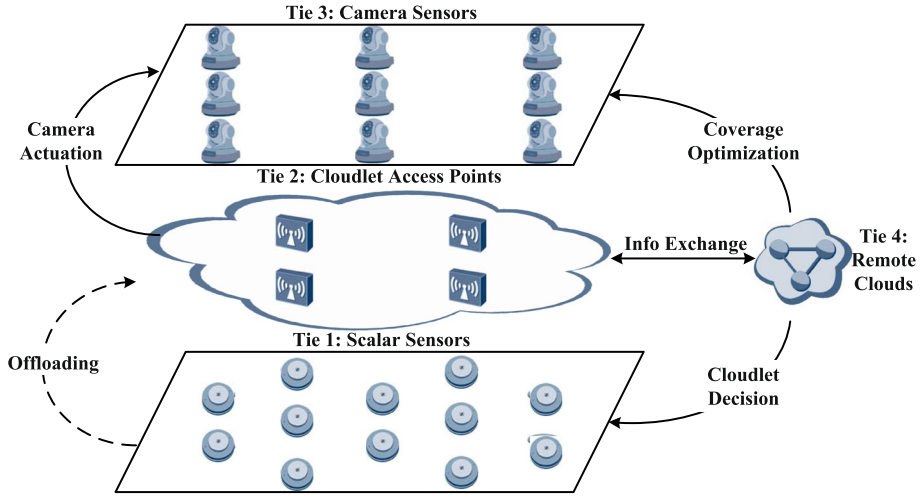

Figure 1 Cloud-assisted multi-tier network.

three control messages, i.e., *ACTUATE*, *OPTIMIZE*, and *UPDATA*. These will be discussed in detail in Section 4.4.

4 Systems components

While the C2ME architecture is intuitive, there are a number of implementation issues to be addressed toward a complete system. In this section, we explain the important components in C2EM. Section 4.1 discusses cloudlets and how to select CAPs to meet QoS requirements. Section 4.2 presents the camera actuation algorithm in the component of the camera selector. Following is the component of the camera optimizer that implements our camera status adjustment algorithm in Section 4.3. We present the component of the camera controller in Section 4.4. Section 4.5 provides comments and discussions on some extensions.

4.1 CAP selector

Offloading computation to local cloudlets through CAPs usually has lower latency to capture the events and has lower energy consumption than to remote clouds. However, the dynamic changes of wireless channel condition contribute negative effects to the reliability and delay of task offloading. Moreover, compared with clouds, cloudlets are limited in computation and storage capacity. When cloudlet workload exceeds a certain threshold, the service response time will be extended.

The goal of the CAP selector is to find the suitable CAPs for scalar sensor offloading, with the objective of minimizing energy consumption while satisfying QoS requirements in offloading delay and reliability. To do this, we implement environmental profilers on cloudlets to monitor server workload, link error ratio, and available bandwidth, which can be used to estimate offloading reliability

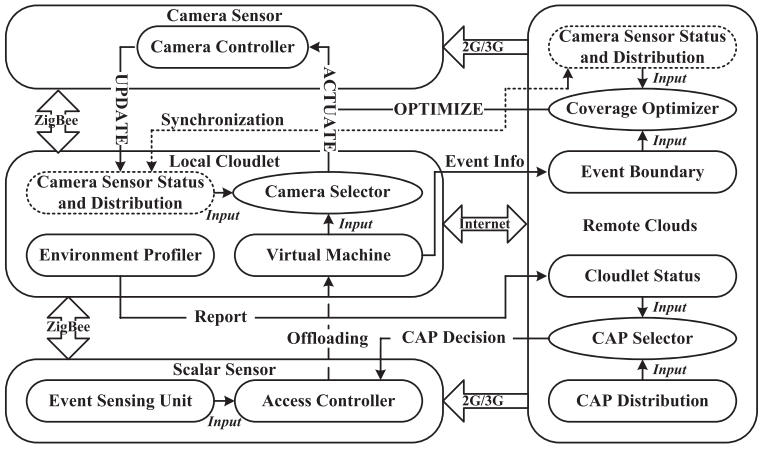

Figure 2 System model.

and service response time. On this basis, we propose an opportunistic service access selection scheme, by exploiting the spatial diversity of the wireless medium to obtain better performance. The optimal CAP set, denoted by δ, is selected for the scalar sensor of ID i according to the following rules.

QoS-aware opportunistic CAP selection (QOCS) rule

$$\begin{aligned} &\text{Given} : \delta \subseteq C \\ &\text{Find} : \delta^* \\ &\text{Minimize} : E_{tx}(l, d_{i\delta}) \\ &\text{Subject to} : \end{aligned} \tag{2}$$

$$d_{i\delta} = \max_{k \in \delta} \text{dist}(i,k) \tag{3}$$

$$R_{i\delta} \geq R_{\text{req}} \tag{4}$$

$$T_{ij} \leq T_{\text{req}}, \forall j \in \delta \tag{5}$$

The optimal CAP set is the cooperative CAP unit that results in the minimum energy consumption under reliability and delay requirements. As shown in (2), the minimization term is the energy consumption for offloading l bits data over a distance $d_{i\delta}$ using our opportunistic service access, where we use a model in [30] for the data communication energy dissipation, and $\text{dist}(i,k)$ is the distance from s_i to a_k (see Equation (3)). Equation (4) is the reliability requirement, where R_{req} is the required reliability. The probability of successfully offloading computation to at least one of the chosen CAP set δ is expressed as

$$R_{i\delta} = 1 - \prod_{j \in \delta} (1 - p_{ij}) \tag{6}$$

Equation (5) is the service delay requirement, where R_{req} is the required delay. Recent studies show that many factors (including server load, available bandwidth, and distance between nodes) affect the service delay [31,32]. This drives us to use a three-dimensional rectangular triple Lagrange interpolation method to compute service delay. To do this, we define interpolation point set $\{(q^{(m)}, b^{(n)}, d^{(k)}) | m \in \{0,1,\ldots,M_j\}, n \in \{0,1,\ldots,N_j\}, k \in \{0,1,\ldots,K_j\}\}$, where $q^{(m)}$ is the m-th value point of server load, $b^{(n)}$ is the n-th value point of available bandwidth, and $d^{(k)}$ is the k-th value point of distance between nodes. On this basis, we express the triple Lagrange interpolation as

$$L(q,b,d) = \sum_{m=0}^{M_j} \sum_{n=0}^{N_j} \sum_{k=0}^{K_j} f(q^{(m)}, b^{(n)}, d^{(k)}) \cdot \ell_{m,n,k}(q,b,d) \tag{7}$$

In Equation (7), the interpolation function

$$\ell_{m,n,k}(q,b,d) = Q_m(q) \cdot B_n(b) \cdot D_k(d) \tag{8}$$

consists of three parts, i.e.,

$$\begin{cases} Q_m(q) = \dfrac{X(q, q^{(m)})}{(q - q^{(m)}) \cdot X'(q, q^{(m)})} \\ B_m(b) = \dfrac{Y(b, b^{(n)})}{(b - b^{(n)}) \cdot Y'(b, b^{(n)})} \\ D_m(d) = \dfrac{Z(d, d^{(k)})}{(d - d^{(k)}) \cdot Z'(d, d^{(k)})} \end{cases} \tag{9}$$

where functions $X(q, q^{(m)})$, $Y(b, b^{(m)})$, $Z(d, d^{(m)})$ are defined as

$$\begin{cases} X(q, q^{(m)}) = \prod_{m=1}^{M_j} (q - q^{(m)}) \\ Y(b, b^{(n)}) = \prod_{n=1}^{N_j} (b - b^{(n)}) \\ Z(d, d^{(k)}) = \prod_{k=1}^{K_j} (d - d^{(k)}) \end{cases} \tag{10}$$

According to above analysis, we estimate T_{ij} as

$$T_{ij} = L(q_i, b_{ij}, dist(i,j)) \tag{11}$$

4.2 Camera selector

The goal of camera selector on cloudlets is to find a camera sensor that has maximal available coverage gain while providing probabilistic guarantee in residual energy. Let Q and $U \subseteq Q$ denote the set of all the deployed camera sensors and the set of actuated camera sensors, respectively. We use $A(\cdot)$ to denote the region area and α_i to denote the ID set of non-actuated camera sensors that have intersection area with s_i.

$$\alpha_i = \{j \in Q \backslash U | A(s_i \cap f_j) > 0\} \tag{12}$$

When a computation task (i.e., an event detected by s_i) arrives, cloudlets will select a suitable camera sensor belonging to α_i, which corresponds to the following rules.

Real-time camera sensor selection (RCSS) rule

$$\begin{aligned} &\text{Given} : j \in \alpha_i \\ &\text{Find} : j^* \\ &\text{Maximize} : \beta_j \\ &\text{Subject to} : \end{aligned} \tag{13}$$

$$\beta_j \geq \theta \tag{14}$$

$$\xi_j < \overline{\xi_j} + (\Delta \xi_j) \cdot \sqrt{\frac{\gamma}{1-\gamma}} \tag{15}$$

As shown in Equation (13), the maximization term is the available coverage gain when c_j is actuated, defined as follows.

$$\beta_j = \frac{A((\Phi \cap f_j) \backslash (\cup_{m \in U} f_m))}{A(\Phi \cap f_j)} \tag{16}$$

Note that scalar sensor nodes can only perceive the existence of the event, i.e., we may not be able to accurately obtain information regarding the event area. As a result, it can be beneficial to use the RCSS rule to get maximum coverage benefits from uncertain information. The condition (14) must be satisfied, where θ is a predefined threshold value.

Denote by e_j the residual energy of camera sensor c_j. In order to balance energy consumption, we provide probabilistic residual energy guarantee. In particular, we use the following method to normalize residual energy of c_j.

$$\xi_j = \begin{cases} \dfrac{e_{\text{mean}} - e_j}{e_{\text{mean}} - e_{\text{min}}}, e_j \le e_{\text{mean}} \\ \dfrac{e_{\text{mean}} - e_j}{e_{\text{max}} - e_{\text{mean}}}, \text{otherwise.} \end{cases} \tag{17}$$

$$e_{\text{min}} = \min_{n \in U} e_n \tag{18}$$

$$e_{\text{max}} = \max_{n \in U} e_n \tag{19}$$

$$e_{\text{mean}} = \frac{1}{|U|} \cdot \sum_{n \in U} e_n \tag{20}$$

It is worth noting that the smaller the value ξ_j, the greater the residual energy of a camera sensor.

In the following, we explain how to achieve the probabilistic residual energy guarantee, i.e., let the probability that the value ξ_j is not less than the threshold value, i.e., ξ_{th} should not be below γ, given by

$$P(\xi_j \le \xi_{\text{th}}) \ge \gamma \tag{21}$$

It can also be expressed as

$$P(\xi_j \ge \xi_{\text{th}}) \le 1 - \gamma \tag{22}$$

According to one-sided Chebyshev's inequality, for a random variable X with mean μ and variance t, it satisfies

$$P(X - \mu \ge t) \le \frac{\sigma^2}{\sigma^2 + t^2} \tag{23}$$

By applying the one-sided Chebyshev's inequality on Equation (22), we have

$$P(\xi_j \ge \xi_{\text{th}}) \le \frac{\left(\Delta\xi_j\right)^2}{\left(\Delta\xi_j\right)^2 + \left(\xi_{\text{th}} - \overline{\xi_j}\right)^2} \tag{24}$$

and

$$\xi_{\text{th}} - \overline{\xi_j} > 0 \tag{25}$$

Based on inequations (24) and (25), we have derived the following two constraints to meet the probabilistic guarantee in inequation (21), i.e.,

$$\frac{\left(\Delta\xi_j\right)^2}{\left(\Delta\xi_j\right)^2 + \left(\xi_{\text{th}} - \overline{\xi_j}\right)^2} \le 1 - \gamma \tag{26}$$

and

$$\xi_{\text{th}} > \overline{\xi_j} \tag{27}$$

If the inequations (26) and (27) hold, the probabilistic guarantee inequation (22) could be met. Inequation (26) can also be expressed as

$$\xi_{\text{th}} \ge \overline{\xi_j} + (\Delta\xi_j) \cdot \sqrt{\frac{\gamma}{1 - \gamma}} \tag{28}$$

According to inequation (28), the threshold value ξ_{th} is defined as

$$\xi_{\text{th}} = \overline{\xi_j} + (\Delta\xi_j) \cdot \sqrt{\frac{\gamma}{1 - \gamma}} \tag{29}$$

Accordingly, the probabilistic guarantee for ξ_j, defined in (16), can be achieved, from which constraint (15) is obtained. ξ_{th} adapts to changes in mean and variance of ξ_j. According to the chosen camera sensors, cloudlets send control messages through CAPs to adjust camera sensor status. In what follows, we will present how to deal with these control messages.

4.3 Coverage optimizer

As an important module implemented on clouds, the camera optimizer is used to compensate the camera selector defects in camera sensor utilization. According to the estimated event boundary, the camera optimizer is responsible for adjusting the state of camera sensors that is associated with the event area in a centralized way, with an objective of improving video sensor utilization, without compromising event coverage quality. To do this, a natural step is to determine the latest event boundary using event detection data stored in database servers. There have been a number of works to determine the location of boundary sensors for an event [33,34]. Each event at a time is identified by a boundary regarding a non-self-intersecting closed polygon, consisting of multiple boundary scalar sensors. We use $\pi(\Lambda)$ to denote the set of boundary vertexes. Let $(x^{(i)}, y^{(i)}) \in \pi(\Lambda)$ be the i-th boundary vertex. The event region area is estimated as

$$A(\Lambda) = \frac{1}{2} \sum_{i=1}^{|\pi(\Lambda)|} (x^{(i)} \cdot y^{(i+1)} - x^{(i+1)} \cdot y^{(i)}) \tag{30}$$

In Equation (30), these vertices are numbered in order of their occurrence along the perimeter of the polygon, and the vertex $(x^{(i)}, y^{(i)})$ is equal to $(x^{(|\pi(\Lambda)|)}, y^{(|\pi(\Lambda)|)})$.

It is worth noting that, due to the non-uniform distribution of scalar sensors, there is a big error between the estimated event area and actual area. Therefore, simply adjusting the camera sensor status in a centralized approach may degrade event coverage quality. Here we present one example to clearly illustrate this insight. In Figure 3, there are six camera sensors and four scalar sensors (black solid points). As we can see, scalar sensors are too sparse to determine the actual event boundary. When using the centralized adjustment approach, camera sensors 3, 4, and 6 will be turned off, and camera sensors 2 and 5 will be actuated, leading to degradation of the event coverage. Therefore, we need to consider how to selectively actuate the coverage optimizer to enhance coverage quality.

Denote by $\eta(\cdot)$ the scalar sensor density within the specified range. We define the following random variable to indicate whether the coverage optimizer should be actuated or not,

$$\omega = \begin{cases} 1, \eta(\Lambda) > \eta(\Phi), \\ 0, \text{otherwise.} \end{cases} \tag{31}$$

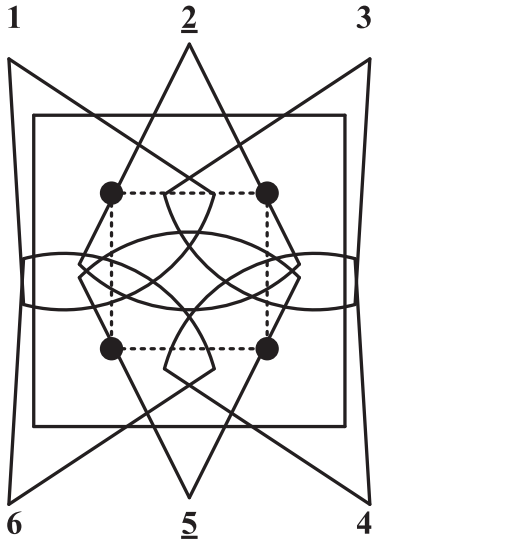

Figure 3 Event boundary example.

If yes, ω is set to be 1, otherwise 0. In Equation (31), $\eta(\Lambda)$ and $\eta(\Phi)$ are computed as

$$\eta(\Lambda) = \frac{|N(\Lambda)|}{A(\Lambda)}, \eta(\Phi) = \frac{|N(\Phi)|}{A(\Phi)} \tag{32}$$

where $N(\Lambda)$ is the number of scalar sensors in Λ.

Denote by $\phi(\Lambda)$ the ID set of camera sensors whose FoV has intersection area with event region Λ.

$$\phi(\Lambda) = \{j \in Q | A(f_j \cap \Lambda) > 0\} \tag{33}$$

We divide each event field into separate sensing regions. For example, as shown in Figure 4, the rectangle sensing field which is totally covered by three camera sensor s has seven sensing regions $\{r_1, r_2, \ldots, r_7\}$. Define $\varpi(\Lambda)$ as the set of sensing regions formed by Λ and FoVs that belongs to $\varpi(\Lambda)$. Let $r_j \in \varpi(\Lambda)$ denote the j-th sensing region.

We use $\rho_{i,j}$ to denote an indicator function of whether r_j can be completely covered by f_i.

$$\rho_{i,j} = \begin{cases} 1, \ A(r_j \cap f_i) = A(r_j), \\ 0, \ \text{otherwise.} \end{cases} \tag{34}$$

Let y_i denote an indicator function of whether camera sensor c_i should be actuated, i.e.,

$$y_i = \begin{cases} 1, \ \text{if } c_i \text{ should be actuated,} \\ 0, \ \text{otherwise.} \end{cases} \tag{35}$$

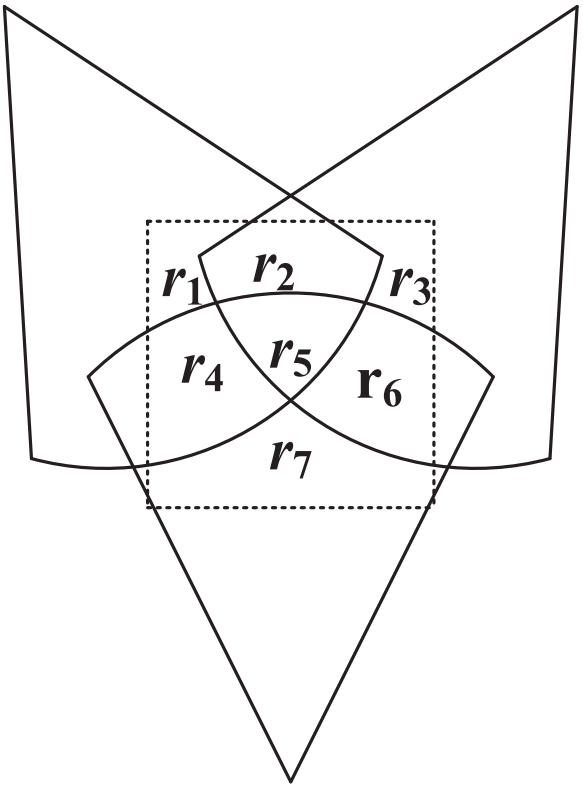

Figure 4 Sensing region example.

The coverage optimization problem can be formulated as the following integer programming.

Sensing-region-based state adjustment (SRSA) problem

$$\underset{\{\phi(\Lambda)\}}{\text{Minimize}} : \sum_{j\in\phi(\Lambda)} y_j - \mu(\Lambda) \tag{36}$$

Subject to :

$$\mu(\Lambda) = \frac{1}{\varsigma} \cdot \min_{i\in\phi(\Lambda)} \{x_j \cdot e_j\} \tag{37}$$

$$\sum_{j\in\phi(\Lambda)} \sum_{i\in\varpi(\Lambda)} y_j \cdot \rho_{i,j} \geq 1 \tag{38}$$

$$y_j \in \{0,1\}, \forall j \in \phi(\Lambda) \tag{39}$$

where ζ is a sufficiently large constant.

In the given formulation, the term of the objective represents the number of camera sensors that should be actuated to achieve an energy-balanced event coverage, which needs to be minimized. Equation (36) consists of $\mu(\Lambda)$, which is used to balance energy consumption. It is not hard to imagine that if $\mu(\Lambda)$ is set to be 0, multiple possible choices may meet coverage requirements. With Equation (37), we choose the camera sensor that has maximal $\mu(\Lambda)$. According to the computation results, clouds send control messages to adjust the status of camera sensors that belongs to $\{j \in Q | x_j \neq y_j\}$. We will explain how to deal with these control messages in the next section.

4.4 Camera controller

The goal of the camera controller on camera sensors is to adjust the camera sensor status according to the received control message. It is worth noting that a number of uncertain factors (e.g., variable service response time and unstable bandwidth) mislead the message handler, resulting in making suboptimal or even inaccurate event coverage. It is not hard to imagine that the control messages from cloudlets and clouds are not synchronous in this environment. One important problem is how to analyze and deal with different control messages from cloudlets and clouds.

The camera sensors, CAPs, and clouds will exchange the messages as detailed in Table 1. The complete pseudo-code for the algorithm is shown in Algorithm 1 and Algorithm 2, corresponding to closed status and actuated status. Each camera sensor runs the algorithms. We will now explain the details of the algorithm based on this pseudo-code.

Table 1 Control messages used in C2EM

Type	Field	Sender	Receiver
ACTUATE	{Camera sensor ID}	CAP	Camera sensor
OPTIMIZE	{Camera sensor ID}	Cloud	Camera sensor
UPDATE	{Camera sensor ID, 1/0(*Turn_On/Turn_Off*)}	Camera sensor	CAP

Algorithm 1: ClosedStatus()

```
1  while c_i=0 do
2      if ACTUATE is received then
3          t_0 ← t_c;
4          c_i ← 1;
5          send(UPDATE);
6      end
7      if OPTIMIZE is received then
8          c_i ← 1;
9          send(UPDATE);
10     end
11 end
```

Algorithm 2: ActuatedStatus()

```
1  while c_i=1 do
2      if OPTIMIZE is received then
3          if t_c − t_0 > φ then
4              c_1 ← 0;
5              send(UPDATE);
6          end
7      end
8  end
```

Algorithm 1 works on closed status. Upon receiving an *ACTUATE* message, a camera sensor will record current time, i.e., t_c, as seen in line 3 (which will be used to indicate whether it should be turned off when receiving an *OPTIMIZE* message later), and then changes its status to *turn_on* so as to capture the events. After that, it broadcasts an *UPDATA* message to local CAPs. On the other hand, upon receiving an *OPTIMIZE* message, it changes its status to *turn_off* and then broadcasts an *UPDATA* message to its neighboring CAPs.

Algorithm 2 works on actuated status. In such status, a camera sensor only needs to deal with *OPTIMIZE* messages. Because offloading computation to local cloudlets through CAPs usually has lower latency that benefits latency-sensitive offloading more, while clouds usually have larger service response time, we deal with different control messages from cloudlets and clouds by assigning a specific interval time. A camera sensor receiving such an *OPTIMIZE* message will carry out a very simple calculation, as seen in line 3, where φ is a threshold value regarding interval time. If the condition is met, a camera sensor will change its status to *turn_off* and send an UPDATA message to local CAPs.

4.5 Discussion

Many event coverage problems are NP-complete [35], and different optimization techniques have been proposed to solve these problems. In general, distributed algorithms or protocols can effectively adapt to changes in network environment and are more preferred for providing the QoS guarantee of timeliness. However, due to limited available information, distributed approaches often provide suboptimal results and therefore offset the benefits to some extent. Instead, although centralized approaches can provide optimal computation results, they are computation-intensive and not applicable for resource-constrained wireless sensor networks. For complex event monitoring in WMSNs, one interesting problem is how to integrate the advantages of both distributed and centralized solutions.

Although the cloudlet tier in C2EM is limited in computation and storage capacity, it usually has lower latency advantage that benefits latency-sensitive offloading more, which corresponds to the advantage of distributed approaches. On the other hand, the cloud tier offers an opportunity to obtain optimal computation results for complex event monitoring, corresponding to the advantage of centralized approaches. Through such two-level optimization, we can make more flexible service for complex event monitoring in WMSNs.

In addition to these, C2EM is a universal cloud-assisted framework. On the one hand, it can support various event converge algorithms and overcome the problem of high computation cost. On the other hand, we can selectively load or enable specific modules according to application requirements.

5 Performance evaluation

This section involves thorough performance analyses and evaluation of the C2EM in simulation methodology. We evaluate the performance of the C2EM through simulation experiments. Section 5.1 describes the evaluation metrics and detailed simulation parameter settings. Section 5.2 presents and analyzes the simulation results.

5.1 Performance metrics

We design and conduct a series of simulation experiments using MATLAB to evaluate the performance of our proposed solution for complex event monitoring. In a monitored region, 250 scalar sensors, 50 camera sensors, and CAPs are deployed randomly, and a sink node is respectively placed in a corner of the field. The sensing directions of the camera sensors are randomly chosen. All camera sensors share the same sensing parameters. Any scalar sensor that detects the event will offload computation to the neighboring CAPs. On the other hand, we consider a WMSN with energy heterogeneity. For initial energy values of camera sensors, we take random values in the range between 0.1 and 0.5 J. The reason why the energy value is initially small is to highlight network performance bottlenecks in a short time, thus allowing to observe the event monitor performance with the passing of time. On the other hand, we assume that scalar sensor nodes have enough energy and can continuously perform event detection. The goal is to accurately evaluate event coverage performance.

In order to reflect the characteristics of the complex events, we place a target event within the monitored region and let it move around according to the random waypoint mobility model and set the pause time to be 0. During movement, the event area changes over time. These actuated camera sensors continuously transmit multimedia data to the sink. The video traces for QCIF frames in [36] are used to simulate the behavior of video data transmission in the simulation, where the video bit rate is set to 64 kbps. To make simulation data more realistic, we assign each cloudlet and cloud with different service response times determined from the RTT statistics in [16]. The default parameters are set in Table 2 unless otherwise specified.

There are many factors that affect the performance of our system, including the initial placement of sensors, the initial placement of CAPs, and the mobility of target event. To eliminate the impact of randomness, 100 scenarios are generated randomly to run for each experiment. In each scenario, the position of sensors and CAPs is uniformly chosen. The presented results are the average of the 100 runs. Let F_n denote the ID set of the camera sensor alive at a certain time point. The following key metrics are used in different scenarios to evaluate C2EM performance regarding system life cycle and event coverage:

Table 2 Parameter settings

Parameters	Value
Monitored region size	100 m × 100 m
Event region size	10 m × 10 m
Average moving speed of target event	5 m/s
Number of scalar sensors	250
Number of camera sensors	50
Scalar sensor sensing radius	1 m
Camera depth of field	15 m
Camera view angle	90°
θ, defined in Equation (14)	10%
γ, defined in Equation (15)	0.5
Transmission rate of actuated camera sensors	64 kbps

- Camera sensor mortality (CSM): the portion of the number of camera sensors that exhausted their energy supply to the total

$$\mathrm{CSM} = \frac{|Q \backslash F_n|}{|Q|} \tag{40}$$

- Available coverage ratio (ACR): the portion of coverage area of all camera sensors alive with regard to the total monitored region

$$\mathrm{ACR} = \frac{A(\Phi \cap (\cup_{m \in F_n} f_m))}{A(\Phi)} \tag{41}$$

- Event coverage ratio (ECR): the expected portion of area of an event covered by actuated camera sensors with regard to its total area

$$\mathrm{ECR} = \frac{A(\Lambda \cap (\cup_{m \in F_n} f_m))}{A(\Lambda)} \tag{42}$$

- FoV utilization (FVU): the expected ratio of area of an event covered by all actuated camera sensors to the total area of FoVs of all actuated camera sensors. The higher the data is, the more redundancy can be eliminated.

$$\mathrm{FVU} = \frac{A(\Lambda \cap (\cup_{m \in F_n} f_m))}{A(\cup_{m \in F_n} f_m)} \tag{43}$$

The focus of our performance evaluation is on several key C2EM function modules that play in the complex event monitoring process. As shown in Table 3, we divided C2EM into three categories, where the difference between these programs lies in enabled functional modules. Because C2EM-1 does not enable the camera selector, the monitoring system has no distributed camera scheduling function. In other words, clouds are responsible for all the work regarding event coverage. C2EM-2 does not enable the coverage optimizer, so the monitoring system does not have centralized camera scheduling function. Cloudlets are responsible for all the work regarding event coverage when using C2EM-2. As a default implementation form of C2EM, C2EM-3 has full cloud-assisted functionality.

5.2 Simulation results

In this section, we assess the performance of different C2EM programs under a variety of conditions.

Figure 5 shows the simulation results on system life cycle, including two performance metrics, i.e., camera sensor mortality and available coverage ratio. If camera sensor redundancy within the monitoring area is small, a single point of death is very likely to lead to decline in available coverage. In practice, we can determine the availability of a monitoring system according to available nodes in mortality or coverage.

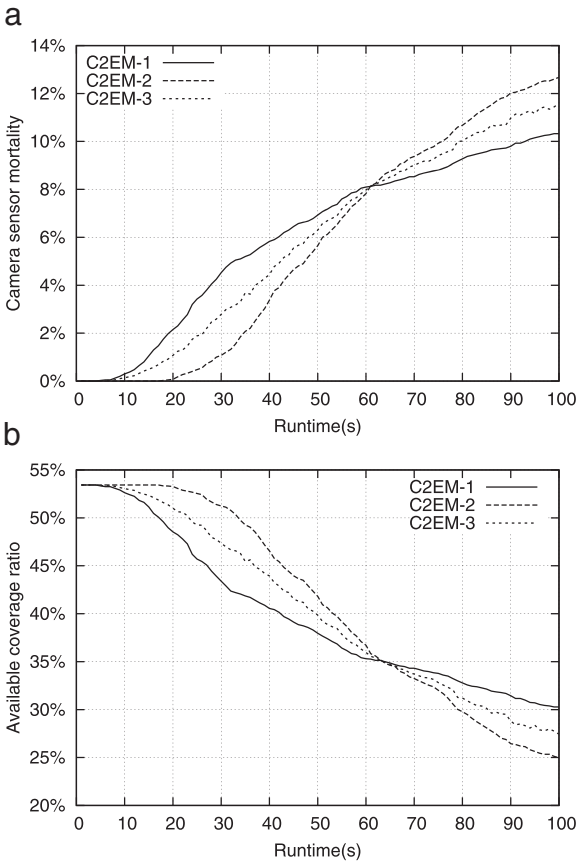

Figure 5 System life cycle. **(a)** Camera sensor mortality. **(b)** Available coverage ratio.

Table 3 Different C2EM programs

Name	Enable camera selector	Enable coverage optimizer
C2EM-1		√
C2EM-2	√	
C2EM-3	√	√

Figure 5a provides camera sensor mortality over runtime. At the early stage, C2EM-1 brings higher camera sensor mortality. This is because the SRSA rule emphasizes precise coverage for an uncertain event area, using a best-effort event coverage approach. In particular, more camera sensors can be used at the stage, increasing the probability that a node is actuated. The low-energy and overused camera sensors will die more quickly. In C2EM-2, the RCSS rule adopts an opportunistic approach to capture an uncertain event, and fewer camera sensors are enabled. Unlike C2EM-1, as there are more camera sensors alive, the RCSS rule can achieve a higher coverage gain, potentially reducing the probability of actuating camera sensors. At the late stage, as the number of available camera sensors decreases, the probability of remaining nodes to be actuated is increased. In C2EM-1,

we consider the effect of the local node density, thus reducing unnecessary data transmissions. Instead, when performing C2EM-2, the system has to enable more nodes to complete event coverage, leading to reduced coverage gain. As shown in Figure 5b, due to increased mortality, the available coverage ratio shows a downward tendency, which is correlated with the result in Figure 5a.

Next we evaluate the simulation results on event coverage, including two performance metrics, i.e., event coverage ratio and FoV utilization. We perform the experiment 100 times to reduce randomness. Because of random placements of camera sensors as well as the dynamic characteristics of the target event, the fluctuations cannot be completely eliminated, but the result data still can reflect the performance of these various C2EM programs.

It can be found in Figure 6a that C2EM-2 achieves the highest event coverage ratio in almost all periods. The main reason is that the RCSS rule can efficiently handle uncertain event information, which provides a benefit to enhance coverage ratio of the blind spot area detected by scalar sensors. In fact, in order to provide high-quality event coverage, it is difficult to reduce the number of enabled nodes. It is worth noting that even at the late stage, C2EM-2 performs better than C2EM-1, indicating that the RCSS rule can effectively adapt to changes in the network environment. In C2EM-1, the SRSA rule cannot accurately cover the actual event area, leading to minimum event coverage. Figure 6b shows FoV utilization over runtime. The higher this utilization is, the more coverage overlaps can be eliminated. As a whole, C2EM-1 provides the maximum FoV utilization. This is because the RCSS algorithm can exploit these overlaps among the FoVs and provide redundancy elimination, thus greatly improving FoV utilization. C2EM-2 provides the lowest FoV utilization. Although it has the advantage of flexibility, the RCSS rule increases the probability of coverage overlaps to some extent. C2EM-3 combines features of both.

We now vary the number of camera sensors at a fixed point in time (in the run to 40 s) to see its effect on camera sensor mortality and available coverage ratio. The results in Figure 7 show that, as expected, the system life cycle increases when the number of camera sensors is

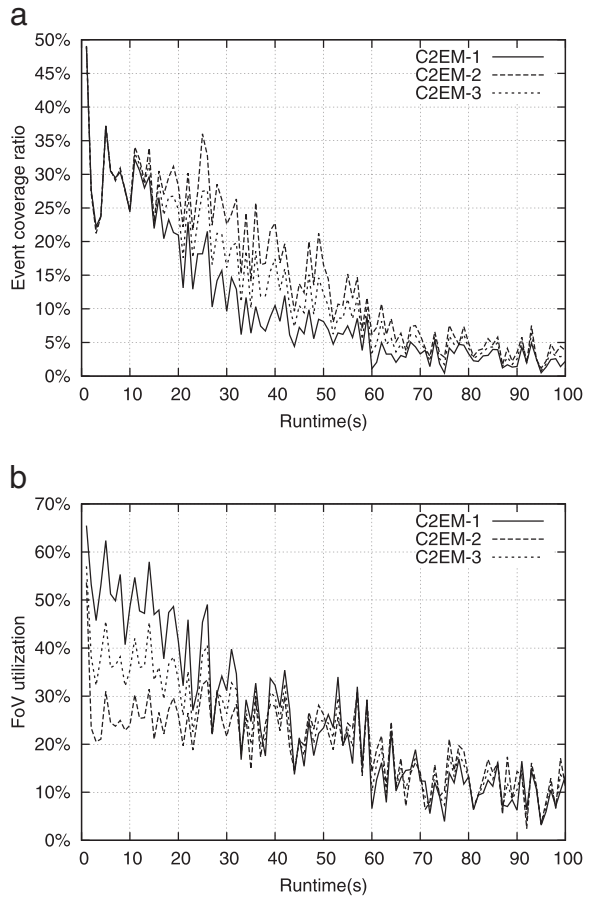

Figure 6 Coverage performance. **(a)** Event coverage ratio. **(b)** FoV utilization.

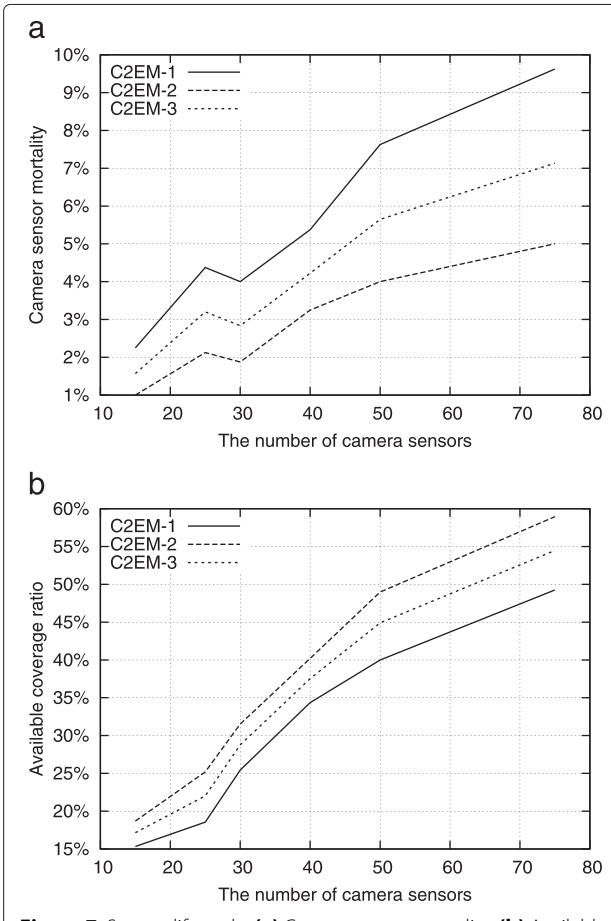

Figure 7 System life cycle. **(a)** Camera sensor mortality. **(b)** Available coverage ratio.

increased. The maximum lifetime is achieved by C2EM-2, then by C2EM-3, followed by C2EM-1. It is worth noting that unlike Figure 5, the differences in performance do not change as the number of camera sensors increases. In other words, the changes in the number of camera sensors will not lead to the performance differences.

Based on these results, we would like to mention that our C2EM architecture has the ability to handle the uncertainties about complex events and provides a reliable and stable complex event monitoring service. In particular, C2EM-3 can be used as a default solution. We can selectively enable specific functional modules, depending on application requirements.

6 Conclusions

Providing an efficient WMSN monitoring service for complex events is a challenging task, in which how to minimize energy consumption providing monitoring quality supports in timeliness and reliability is a major concern. However, apart from the limited bandwidth and energy, the high computation burden caused by complex events is another important problem, because sensors are constrained in computation capacity, while complex event monitoring is computation-intensive. To address this problem, we propose C2EM, a cloud-assisted complex event monitoring architecture, in consideration of computation offloading reliability, service response time, coverage efficiency, and energy efficiency. The C2EM takes advantages of both distributed and centralized coverage approaches. Through computation division, most complex computations and network environment profilers are executed on cloudlets or clouds. Sensors only need to carry out very simple operations. Our simulation results demonstrate that C2EM can provide more flexible service for complex event monitoring with optimized energy efficiency, desirable event coverage quality, and potential adaptability to the dynamics of complex events.

Our future work will focus on developing the QoS-aware module for cloud-based video processing and on utilizing multiple CAPs to accommodate video applications with overlarge offload data size. We will also verify the effectiveness of our event monitoring algorithms and then implement the C2EM architecture in a real-world environment.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

The authors would like to thank the anonymous reviewers who provided excellent and thorough feedback that improved the quality of this paper. The authors gratefully acknowledge the support and financial assistance provided by the National Natural Science Foundation of China under Grant Nos. 60673185 and 61073197, the Scientific & Technological Support Project (Industry) of Jiangsu Province under No. BE2011186, the Prospective Study of Future Network of Jiangsu Province under No. BY2013095-4-09, the Key Lab Program of Broadband Wireless Communication and Sensor Network Technology (Nanjing University of Posts and Telecommunications), and Ministry of Education under Grant No. NYKL201304.

Author details

[1]College of Electronics and Information Engineering, Nanjing Tech University, No. 30, PuZhu Road (South), Pukou District, 211816 Nanjing, China. [2]College of Computer Science and Engineering, Nanjing University of Science and Technology, Xiaolingwei Street 200, Xuanwu District, 210094 Nanjing, China. [3]Key Lab of Broadband Wireless Communication and Sensor Network Technology (Nanjing University of Posts and Telecommunications), Ministry of Education, XinMofan Road 66, 210003 Nanjing, China.

Received: 21 November 2014 Accepted: 31 March 2015
Published online: 30 April 2015

References

1. A Newell, K Akkaya, Distributed collaborative camera actuation for redundant data elimination in wireless multimedia sensor networks. Ad Hoc Netw. **9**(4), 514–527 (2011). doi:10.1016/j.adhoc.2010.08.003
2. C Li, J Zou, H Xiong, CW Chen, Joint coding/routing optimization for distributed video sources in wireless visual sensor networks. IEEE Trans. Circuits Syst. Video Technol. **21**(2), 141–155 (2011). doi:10.1109/TCSVT.2011.2105596
3. H Shen, G Bai, Z Tang, L Zhao, QMOR: QoS-aware multi-sink opportunistic routing for wireless multimedia sensor networks. Wireless Pers. Commun. **75**(2), 1307–1330 (2014). doi:10.1007/s11277-013-1425-0
4. Y He, I Lee, L Guan, Distributed algorithms for network lifetime maximization in wireless visual sensor networks. IEEE Trans. Circuits Syst. Video Technol. **19**(5), 704–718 (2009). doi:10.1109/TCSVT.2009.2017411
5. VP Munishwar, NB Abu-Ghazaleh, Coverage algorithms for visual sensor networks. ACM Trans. Sensor Netw. (TOSN). **9**(4), 1–36 (2013). doi:10.1145/2489253.2489262
6. M Schwager, BJ Julian, M Angermann, D Rus, Eyes in the sky: decentralized control for the deployment of robotic camera networks. Proc. IEEE. **99**(9), 1541–1561 (2011). doi:10.1109/JPROC.2011.2158377
7. VP Munishwar, Abu-Ghazaleh N B, Design of multimedia surveillance systems. ACM Trans. Multimedia Comput. Commun. Appl. (TOMM). **5**(3), 1–25 (2009). doi:10.1145/1556134.1556140
8. B Hore, J Wickramasuriya, S Mehrotra, N Venkatasubramanian, D Massaguer, in *IEEE International Conference on Pervasive Computing and Communications (PERCOM)*. Privacy-preserving event detection in pervasive spaces (Galveston, TX, USA, 2009), pp. 1–10. doi:10.1109/PERCOM.2009.4912772. http://dx.doi.org/10.1109/PERCOM.2009.4912772
9. A Czarlinska, D Kundur, Reliable event-detection in wireless visual sensor networks through scalar collaboration and game-theoretic consideration. IEEE Trans. Multimedia. **10**(5), 675–690 (2008). doi:10.1109/TMM.2008.922775
10. Y Keung, B Li, Q Zhang, in *Proceedings of the Eleventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. The intrusion detection in mobile sensor network (Chicago, Illinois, USA, 2010), pp. 11–20. doi:10.1145/1860093.1860096. http://doi.acm.org/10.1145/1860093.1860096
11. N Fernando, SW Loke, W Rahayu, Mobile cloud computing: a survey. Future Generation Comput. Syst. **29**(1), 84–106 (2013). doi:10.1016/j.future.2012.05.023
12. M Satyanarayanan, P Bahl, R Caceres, N Davies, The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2009). doi:10.1109/MPRV.2009.82
13. I Kelényi, JK Nurminen, in *Proceedings of the 7th IEEE Conference on Consumer Communications and Networking Conference (CCNC)*. Cloudtorrent-energy-efficient bittorrent content sharing for mobile devices via cloud services (Las Vegas, NV, USA, 2010), pp. 646–647. doi:10.1109/CCNC.2010.5421712
14. B-G Chun, S Ihm, P Maniatis, M Naik, A Patti, in *Proceedings of the Sixth Conference on Computer Systems (EuroSys)*. Clonecloud: elastic execution between mobile device and cloud (Salzburg, Austria, 2011), pp. 301–314. doi:10.1145/1966445.1966473. http://doi.acm.org/10.1145/1966445.1966473
15. G Huerta-Canepa, D Lee, in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (MCS)*. A virtual

cloud computing provider for mobile devices (San Francisco, CA, USA, 2010), pp. 1–5. doi:10.1145/1810931.1810937. http://doi.acm.org/10.1145/1810931.1810937
16. E Cuervo, A Balasubramanian, Cho D-k, A Wolman, S Saroiu, R Chandra, P Bahl, in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*. MAUI: making smartphones last longer with code offload (San Francisco, CA, USA, 2010), pp. 49–62. doi:10.1145/1814433.1814441. http://doi.acm.org/10.1145/1814433.1814441
17. C Shi, K Habak, P Pandurangan, M Ammar, M Naik, E Zegura, in *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. COSMOS: computation offloading as a service for mobile devices (Philadelphia, PA, USA, 2014), pp. 287–296. doi:10.1145/2632951.2632958. http://doi.acm.org/10.1145/2632951.2632958
18. A Fahim, A Mtibaa, KA Harras, in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking (MobiCom)*. Making the case for computational offloading in mobile device clouds (Miami, Florida, USA, 2013), pp. 203–205. doi:10.1145/2500423.2504576. http://doi.acm.org/10.1145/2500423.2504576
19. X Wang, M Chen, TT Kwon, L Yang, V Leung, AMES-cloud: a framework of adaptive mobile video streaming and efficient social video sharing in the clouds. IEEE Trans. Multimedia. **15**(4), 811–820 (2013). doi:10.1109/TMM.2013.2239630
20. P Simoens, Y Xiao, P Pillai, Z Chen, K Ha, M Satyanarayanan, in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*. Scalable crowd-sourcing of video from mobile devices (Taipei, Taiwan, 2013), pp. 139–152. doi:10.1145/2462456.2464440. http://doi.acm.org/10.1145/2462456.2464440
21. A Alamri, WS Ansari, MM Hassan, MS Hossain, A Alelaiwi, MA Hossain, A survey on sensor-cloud: architecture, applications, and approaches. Int. J. Distributed Sensor Netw. **2013** (2013). doi:10.1155/2013/917923
22. V Casola, Benedictis A D, M Rak, G Aversano, U Villano, in *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*. An SLA-based approach to manage sensor networks as-a-service, vol. 1 (Bristol, UK, 2013), pp. 191–197. doi:10.1109/CloudCom.2013.33
23. N Mitton, S Papavassiliou, A Puliafito, K Trivedi, Combining cloud and sensors in a smart city environment. EURASIP J. Wireless Commun. Netw. **2012**(1) (2012). doi:10.1186/1687-1499-2012-247
24. R Piyare, S Park, SY Maeng, SH Park, SC Oh, SG Choi, HS Choi, SR Lee, in *International Conference on ICT Convergence (ICTC)*. Integrating wireless sensor network into cloud services for real-time data collection (Jeju Island, Korea, 2013), pp. 752–756. doi:10.1109/ICTC.2013.6675470
25. L Filipponi, A Vitaletti, G Landi, V Memeo, G Laura, P Pucci, in *International Conference on Sensor Technologies and Applications (SENSORCOMM)*. Smart city: an event driven architecture for monitoring public spaces with heterogeneous sensors (Venice, Italy, 2010), pp. 281–286. doi:10.1109/SENSORCOMM.2010.50
26. M Shiraz, A Gani, RH Khokhar, R Buyya, A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. IEEE Commun. Surv. Tutor. **15**(3), 1294–1313 (2013). doi:10.1109/SURV.2012.111412.00045
27. L Yang, J Cao, Y Yuan, T Li, A Han, A Chan, A framework for partitioning and execution of data stream applications in mobile cloud computing. ACM SIGMETRICS Perform. Eval. Rev. **40**(4), 23–32 (2013). doi:10.1145/2479942.2479946
28. Y Xu, S Mao, A survey of mobile cloud computing for rich media applications. IEEE Wireless Commun. **20**(3), 46–53 (2013). doi:10.1109/MWC.2013.6549282
29. G Mao, B Fidan, B Anderson, Wireless sensor network localization techniques. Comput. Netw. **51**(10), 2529–2553 (2007). doi:10.1016/j.comnet.2006.11.018
30. WB Heinzelman, AP Chandrakasan, H Balakrishnan, An application-specific protocol architecture for wireless microsensor networks. IEEE Trans. Wireless Commun. **1**(4), 660–670 (2002). doi:10.1109/TWC.2002.804190
31. J Li, K Bu, X Liu, B Xiao, in *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing (MCC)*. Enda: Embracing network inconsistency for dynamic application offloading in mobile cloud computing (Hong Kong, China, 2013), pp. 39–44. doi:10.1145/2491266.2491274. http://doi.acm.org/10.1145/2491266.2491274
32. X Xiang, C Lin, X Chen, Energy-efficient link selection and transmission scheduling in mobile cloud computing. IEEE Wireless Commun. Lett. **3**(2), 153–156 (2014). doi:10.1109/WCL.2013.122113.130825
33. K Ren, K Zeng, W Lou, in *IEEE Global Telecommunications Conference (GLOBECOM)*. Fault-tolerant event boundary detection in wireless sensor networks (San Francisco, CA, USA, 2006), pp. 1–5. doi:10.1109/GLOCOM.2006.514
34. M Ding, D Chen, K Xing, X Cheng, in *Proceedings of IEEE INFOCOM*. Localized fault-tolerant event boundary detection in sensor networks, vol. 2 (Miami, FL, USA, 2005), pp. 902–913. doi:10.1109/INFCOM.2005.1498320
35. B Wang, Coverage problems in sensor networks: a survey. ACM Comput. Surv. (CSUR). **43**(4), 32–13253 (2011). doi:10.1145/1978802.1978811
36. P Seeling, M Reisslein, B Kulapala, Network performance evaluation using frame size and quality traces of single-layer and two-layer video: a tutorial. IEEE Commun. Surv. Tutor. **6**(3), 58–78 (2004). doi:10.1109/COMST.2004.5342293